

S10_L3

Unit 3 - CS0424

MATTEO BELTRAMI MARZOLINI
CYBEREAGLES

Giorno 3 – Assembly x86

TRACCIA

Nella lezione teorica del mattino, abbiamo visto i fondamenti del linguaggio Assembly. Dato il codice in Assembly per la CPU x86 allegato qui di seguito, identificare lo scopo di ogni istruzione, inserendo una descrizione per ogni riga di codice. Ricordate che i numeri nel formato 0xYY sono numeri esadecimali. Per convertirli in numeri decimali utilizzate pure un convertitore online, oppure la calcolatrice del vostro computer (per programmatori).

```
0x00001141 <+8>:  mov  EAX,0x20
0x00001148 <+15>:  mov  EDX,0x38
0x00001155 <+28>:  add   EAX,EDX
0x00001157 <+30>:  mov  EBP,EAX
0x0000115a <+33>:  cmp  EBP,0xa
0x0000115e <+37>:  jge  0x1176 <main+61>
0x0000116a <+49>:  mov  eax,0x0
0x0000116f <+54>:  call 0x1030 <printf@plt>
```

SVOLGIMENTO

0x00001141 <+8>: mov EAX, 0x20

Sposta il valore esadecimale

0x20 $(2 \times 16^1 + 0 \times 16^0) = (2 \times 16) + (0 \times 1) = \mathbf{32}$ in decimale

nel **registro EAX**.

0x00001148 <+15>: mov EDX, 0x38

Sposta il valore esadecimale

0x38 $(3 \times 16^1 + 8 \times 16^0) = (3 \times 16) + (8 \times 1) = 48 + 8 = \mathbf{56}$ in decimale

nel **registro EDX**

0x00001155 <+28>: add EAX, EDX

Aggiunge il valore contenuto nel **registro EDX** (56 in decimale) **al valore contenuto nel registro EAX** (32 in decimale), **memorizzando** il risultato in **EAX**.

Dopo questa operazione, **EAX conterrà il valore 0x58** (38+20)

$(5 \times 16^1 + 8 \times 16^0) = (5 \times 16) + (8 \times 1) = 80 + 8 = \mathbf{88}$ in decimale

0x00001157 <+30>: mov EBP, EAX

Muove il valore contenuto nel **registro EAX** (88 in decimale) **nel registro EBP**.

0x0000115a <+33>: cmp EBP, 0xa

Confronta il valore contenuto nel **registro EBP (88)** con il **valore esadecimale 0xa**

a in esadecimale **equivale a 10 in decimale**

Questa istruzione modifica i flag nel registro EFLAGS in base al risultato del confronto.

0x0000115e <+37>: jge 0x1176 <main+61>

0x0000115e <+37> Questo è l'indirizzo di memoria dove si trova l'istruzione jge.

Se il valore contenuto in **EBP è maggiore o uguale a 0xa (10 in decimale)**, esegue un **salto all'indirizzo di memoria 0x1176**, l'indirizzo di destinazione dove saltare se la condizione è soddisfatta, che corrisponde all'indirizzo esadecimale specifico **<main+61>** (61 byte dall'inizio della funzione main).

0x0000116a <+49>: mov eax, 0x0

Sovrascrive il valore **0x0** (0 in decimale) **nel registro EAX**.

0x0000116f <+54>: call 0x1030 printf@plt

0x0000116f <+54> Questo è l'indirizzo di memoria dove si trova l'istruzione call nel codice binario

Chiama la **funzione situata** all'indirizzo **0x1030**, che corrisponde a **printf@plt**. Quindi il controllo viene passato a printf per eseguire la stampa.